Extended CUG for Free Word Order Languages and its Efficient Implementation within an IDA Architecture

Werner Winiwarter

Institute of Applied Computer Science & Information Systems
Dept. of Information Engineering, University of Vienna
Liebiggasse 4/3, A-1010 Wien, Austria
EMAIL: ww@ifs.univie.ac.at

Abstract

This paper is focused on the extension of Categorial Unification Grammar (CUG) in order to achieve a compact grammatical representation scheme for the syntactic analysis of free word order languages. The resulting grammar was applied to the implementation of an efficient parser within a German natural language interface to a deductive database by making use of the Integrated Deductive Approach in which the interface constitutes a fully integrated part of the database system.

1. Introduction

The Integrated Deductive Approach (IDA) is an architecture which does not design the interface as loosely coupled filter separated from the database but on the contrary integrates it in the database management system (DBMS) itself so that the dictionary of functional terms becomes part of the database and the complete natural language analysis is performed by the logic language provided by deductive databases [1]. This comprises the two main advantages that there are no more breaks in homogeneity as concerns the mapping of the final semantic representation onto the database scheme and no more inaccuracies with regard to the treatment of unknown input words.

We applied the IDA architecture to the design and implementation of a German natural language interface to a production planning and control system (PPC). Due to the high complexity of the application domain exceeding the capabilities of traditional relational DBMS, the PPC was modelled by use of the deductive database LDL implemented at MCC [2, 3].

The dictionary used in the natural language interface is built out of canonical forms as hierarchical deductive database in LDL. This allows to assign all morphological, syntactic, and semantic features to the appropriate level of abstraction (e.g. special derivations, word stems, endings, word categories etc.) by making use of inheritance rules [4].

The aim of syntactic analysis within natural language interfaces should not be the coverage of the complete language, especially not of all those exotic phenomena possessing only linguistic evidence but no practical relevance. Whilst the

generality is therefore on the one hand restricted in comparison with other applications of natural language processing, it has to provide on the other hand important extensions indispensable for effective information retrieval [5]:

- tolerance as concerns ungrammatical sentences
- correct interpretation of incomplete sentences
- processing of unknown words

The paper is structured as follows. After a discussion of related work in Section 2 we develop the required formal framework of syntactic analysis in Section3. Finally, Section 4 deals with the implementation of the parser within IDA.

2. Background and related work

Categorial Grammar (CG) is an unconventional grammar theory which assigns all grammar rules to the dictionary entries, making any additional explicit grammar superfluous. There exist two kinds of categories: basic categories (e.g. S, N) and complex categories (e.g. NP/N, S\NP). The original theory [6] consists of only two combinatory rules for the formation of complex categories (A, B representing grammatical categories):

- Forward functional application: $A/B\ B \rightarrow A$
- Backward functional application: $B\ A\backslash B \rightarrow A$

The two combinatory rules were extended by Steedman [7,8] by four rules for functional composition and type raising resulting in Combinatory Categorial Grammar (CCG) (see also [9, 10]):

- Forward functional composition: $A/B\ B/C \rightarrow A/C$
- Backward functional composition: $B\backslash C\ A\backslash B \rightarrow A\backslash C$
- Type raising: $B \rightarrow A/(A\backslash B)$, $B \rightarrow A\backslash(A/B)$

As an additional extension variable categories have been proposed [11, 12]. This powerful generative capacity has been applied to cover some important non-canonical natural language constructions like wh-extraction or nonconstituent conjunction (e.g. see [13, 14]). The other side of the coin is that parsing of CCG as such has been turned out to be inefficient leading to spurious ambiguity [15]. Therefore, a lot of work was done to improve the performance of CCG parsers, e.g. compiling the grammar in a predictive form [16, 17], normal-form based parsing [18, 19] or lazy chart parsing techniques [20]. Vijay-Shanker and Weir have proposed the only polynomial parsing scheme so far by using stacking machinery [21, 22].

By adding features, local registers, and unification operators, CCG was extended to Categorial Unification Grammar (CUG) introduced by Uszkoreit [23]. Besides of syntax analysis, morphology [24, 25], natural language generation [26], and speech processing [27] have been proposed as application fields of categorial grammars. Also some work was done on the treatment of modifiers,

specifiers, and quantifiers [28, 29]. In order to solve the problem of constituent transfer without using type raising and functional composition, the technique of gap-threading has been proposed [30, 31].

As formal framework for the syntactic analysis within IDA we have selected CUG because of two main reasons:

- the availability of a powerful hierarchical dictionary which makes it possible to assign the grammar rules to the appropriate level of abstraction
- the bottom-up parsing strategy which is in conformity with LDL semantics and makes it possible to analyse incomplete or ill-formed sentences in a natural way

Since CUG as such is not applicable to languages with free word order like German and the solutions proposed and presented above all lack of declarative expressiveness, we chose an alternative approach of extending CUG by a formalism adapted from the ID/LP rules of GPSG which have been proven adequate for this purpose [32, 33].

3. Grammar Formalism

We changed the original notation for the two combinatory rules of functional application in order to be able to present the proposed extensions in a consistent way:

- C: A ← /B

  If category C is directly followed by B, then it can be transformed to A (forward functional application)

- C: A ← \B

  If category C is directly preceded by B, then it can be transformed to A (backward functional application)

We extended the Categorial Unification Grammar by the following concepts. Each extension is clarified by use of an example rule, its verbalisation and the application of the rule to an example phrase, e.g.:

PREP: PP ← /NP

A prepositional phrase consists of a preposition directly followed by a noun phrase.

[PREP: auf, NP: die Maschine] → [PP: auf die Maschine]
(to the machine)

❶ Syntactic feature restrictions can be added in brackets to the categories.

This filter function increases significantly the selectivity of the parser during unification.

NP[-unb]: NP ← /NP[+unb]

An unknown phrase (by default assigned to basic category NP) can be joined with a known noun phrase to form a combined one.

[NP: die Maschine, NP: 7] → [NP: die Maschine 7] (the machine 7)

❷ New symbols > for indirect precedence and < for indirect succession.

This covers cases of long distance dependencies where several phrases can be shifted between the two concerned categories.

TRVERB[+sep, -inf, -part]: TRVERB ← <VPREF

Separable verb prefixes can take positions far behind the verb stem if the verb form is neither infinitive nor participle.

[TRVERB: führe, NOUN: Auftrag, VPREF: durch] →
[TRVERB: führe durch, NOUN: Auftrag] (execute order)

❸ More than one category can be used at the right side of a rule.

This removes the severe restriction that in a single derivation step only adjacent categories can be applied to the derivation of a new category. The several categories are applied from left to right.

NOUN: NP ← \ADJ \ART

A noun is transformed to a noun phrase if it is directly preceded by an adjective and an article.

[ART: der, ADJ: neue, NOUN: Gehalt] → [NP: der neue Gehalt]
(the new salary)

❹ Introduction of the asterisk symbol indicating as usual zero or more repetitions.

This extension is introduced in order to capture recursive constructs of phrases.

NOUN: NP ← \ADJ* \ART

A noun phrase consists of an article, several optional adjectives, and a noun.

[ART: der, ADJ: neue, ADJ: monatliche, NOUN: Gehalt] →
[NP: der neue monatliche Gehalt] (the new monthly salary)

❺ Enclosing of optional categories in parenthesis.

Optional categories reduce the number of required grammar rules and increase at the same time the clearness of representation.

NOUN: NP ← \ADJ* (\ART)

A noun, directly preceded by several optional adjectives and an optional article, generates a noun phrase.

[ADJ: neuer, NOUN: Gehalt] → [NP: neuer Gehalt] (new salary)

❻ No function symbol in front of a category to indicate free word order.

In this situation it is only necessary that the category is present in the sentence but no further conditions on its position are made.

TRVERB[+imp]: IC ← NP[+akk] PP*

A transitive verb with mood imperative accompanied by a noun phrase with case accusative and several optional prepositional phrases creates an imperative clause, the sequence of the three components is arbitrary.

[TRVERB: storniere, PP: für den Kunden Maier, NP: den letzten Auftrag]
→ [IC: storniere den letzten Auftrag für den Kunden Maier]
(cancel the last order for the client Maier)

4. Implementation

The grammar rules are inserted as arguments to the dictionary at the appropriate level of abstraction. This grammatical argument possesses the following format:

{(NewCat, Restrict, RightSide, Priority)}

NewCat	derived category
Restrict	syntactic restrictions
RightSide	right side of grammar rule
Priority	priority value which determines the application order of the rules

The right side of the grammar rule is mapped to a list of categories:

[(Cat, Restrict, Sequence, Occurrence)]

Cat	category
Restrict	syntactic restrictions
Sequence	sequence condition
Occurrence	occurrence condition

Possible values for Occurrence are erforderlich (required), optional, and '*', for Sequence: ' ', '/', '\', '>', and '<', e.g.:

TRVERB[+imp]: IC ← NP[+akk] PP* ⇒
{(ic, {('+', imp)}, [(np, {('+', akk)}, ' ', erforderlich), (pp, { }, ' ', '*')], 1)}

NOUN: NP ← \ADJ \ART ⇒
{(np, { }, [(adj, { }, '\', '*'), (art, { }, '\', optional)], 5)}.

The parsing of an input sentence is performed in the following way. First, based on the result of lexical analysis a basic category is assigned to each word resulting in an appropriate list representation. Then, the grammar rules are applied to this list by use of a bottom-up strategy. The syntactic features of the combined structures are unified at each step leading to a significant reduction of derivations.

As already mentioned the decision which rule is applied next is not arbitrary but is determined by the priority values stored in the dictionary. The deliberate choice of these values is of crucial importance to the efficiency of the parser in that the additional effort for backtracking and trying other interpretations is minimised. Figure 1 shows an example segment of the LDL code which represents the top level of syntactic analysis, it recursively produces all applicable rules and selects the one with the highest priority value for derivation until no more rules can be applied.

```
analys(Liste1, Liste3) ←                recursive rule for the syntactic analysis of an input
                                        sentence (Liste1 ... input list, Liste3 ... output list)
   regel(Liste1, Regeln),               generation of all applicable rules
   Regeln ~= { },                       rule set not empty
   aggregate(maxpri, Regeln, Best),     determination of rule with the highest priority
   Best = (Liste2, _),                  new list after application of grammar rule
   analys(Liste2, Liste3).              next step of recursion
analys(Liste, Liste) ←                  exit rule
   regel(Liste, { }).                   triggers if no more rules can be applied

regel(Liste, Regeln) ←                  produces all applicable rules to input list Liste
   regel2(1, Liste, Liste, Regeln).     initial call of recursive generation rule

regel2(I, [X|Rest], Liste, Regeln) ←    I ... list position, X ... processed category,
                                        Rest ... rest of list
   genregel(I, X, Liste, Regeln2),      generates set of all applicable rules for category X
   I2 = I + 1,                          incrementing list position
   regel2(I2, Rest, Liste, Regeln3),    next step of recursion
   union(Regeln2, Regeln3, Regeln).     resulting rule sets are merged
regel2(_, [ ], _, { }).                 exit rule
```

Figure 1: LDL code segment of syntactic analysis

The second example code in Figure 2 checks the applicability of the right side of a single grammar rule to a specific category. Recursively, each constituent of the right side is checked against the list members. If the test holds true for the right side, the concerned list members are replaced by the new derived syntactic category.

```
analys(Pos1, Regelliste, List1, List3) <-     recursive analysis of single grammar rule
                                              Pos1 ... position of considered list entry
                                              List1... input list, List3 ... output list
   Regelliste = [Eintrag|Rest],               Regelliste ... right side of grammar rule
   analys2(Pos1, Eintrag, List1, List2),      analysis of first entry of right side
   analys(Pos1, Rest, List2, List3).          analysis of rest of right side
analys( _, [ ], List, List).                  exit rule

analys2(Pos1, (Kat, Restr,                    analysis of single entry of right side
      Seq, Occ), List1, List3) <-             Kat ... syntactic category to be searched
                                              Restr ... syntactic restrictions
                                              Seq ... sequence condition
                                              Occ ... occurrence condition
   lmember(Kat, Restr, Pos2, List1),          membership test yielding position in list
   if(Seq = '<'                               checking of sequence conditions
   then Pos1 < Pos2),
   ...
   entferne(Kat, List1, List2),               removal of entry from input list
   if(Occ = '*'                               if occurrence is repetitive then
   then  analys2(Pos1, (Kat, Restr,           recursive application of rule
      Seq, Occ), List2, List3)
   else  List3 = List2).
analys2( _, (Kat, Restr, _, optional), L, L) <- analysis rule for optional occurrence
   ~lmember(Kat, Restr, _, L).                   if concerned category is absent
analys2( _, (Kat, Restr, _, '*'), L, L) <-    exit rule for repetitive occurrence
   ~lmember(Kat, Restr, _, L).
```

Figure 2: Test of the applicability of the right side of a grammar rule

Finally, Figure 3 shows a simplified example (leaving out of consideration the unified features) of the individual steps of syntactic analysis of an example input sentence.

Example sentence:
Füge den neuen Mitarbeiter Max Huber mit Anfangsgehalt 20000 hinzu !
(Insert the new worker Max Huber with initial salary 20000 !)

Basic categories:
[TRVERB, ART, ADJ, NOUN, NP, PREP, NOUN, NP, VPREF]
[Füge, den, neuen, Mitarbeiter, Max Huber, mit, Anfangsgehalt, 20000, hinzu]

Analysis:

1) TRVERB[+sep, -inf, -part]: TRVERB ← <VPREF
[TRVERB, ART, ADJ, NOUN, NP, PREP, NOUN, NP]
[Füge hinzu, den, neuen, Mitarbeiter, Max Huber, mit, Anfangsgehalt, 20000]

2) NOUN: NP ← \ADJ* (\ART)
[TRVERB, NP, NP, PREP, NOUN, NP]
[Füge hinzu, den neuen Mitarbeiter, Max Huber, mit, Anfangsgehalt, 20000]

3) NOUN: NP ← \ADJ* (\ART)
[TRVERB, NP, NP, PREP, NP, NP]
[Füge hinzu, den neuen Mitarbeiter, Max Huber, mit, Anfangsgehalt, 20000]

4) NP[-unb]: NP ← /NP[+unb]
[TRVERB, NP, PREP, NP, NP]
[Füge hinzu, den neuen Mitarbeiter Max Huber, mit, Anfangsgehalt, 20000]

5) NP[-unb]: NP ← /NP[+unb]
[TRVERB, NP, PREP, NP]
[Füge hinzu, den neuen Mitarbeiter Max Huber, mit, Anfangsgehalt 20000]

6) PREP: PP ← /NP
[TRVERB, NP, PP]
[Füge hinzu, den neuen Mitarbeiter Max Huber, mit Anfangsgehalt 20000]

7) TRVERB[+imp]: IC ← NP[+akk] PP*
[IC]
[Füge hinzu den neuen Mitarbeiter Max Huber mit Anfangsgehalt 20000]

Figure 3: Example of syntactic analysis

Of course, the main task of syntactic analysis within a natural language interface is not to derive the syntactic correctness of an input sentence but to construct its syntactic structure in parallel. For the sentence shown in Figure 3 this structure looks as displayed in Figure 4 (not showing morphological and syntactic features in detail). The symbol `c` stands for a derived category whereas `s` signifies basic categories.

```
[(ic, c, [
        (trverb, c, [
                (trverb, s, fuege),
                (vpref, s, hinzu)
        ]),
        (np, c, [
                (np, c, [
                        (noun, s, mitarbeiter)
                        (adj, s, neuen)
                        (art, s, den)
                        ]),
                (np, s, 'Hubert Maier')
                ])
        ]),
        (pp, c, [
                (prep, s, mit),
                (np, c, [
                        (np, c, [
                                (noun, s, 'Anfangsgehalt')
                        ]),
                        (np, s, 20000)
                ])
        ])
])]
```

Figure 4: Example of syntactic structure

5. Conclusion

The current state of our work is as follows. The implementation of the PPC and the morphological, lexical, and syntactic analysis of our natural language interface is completed, the remaining components are subject of ongoing research. The components of the prototype implemented so far were tested extensively by use of 1000 realistic example sentences obtained by questionnaires. This resulted in a compact LDL database capable of dealing with all occurring syntactic phenomena in a concise and flexible way. By using as hardware configuration a SUN SPARC 10 we achieved response times of less than two seconds even for complex user queries.

This performance will be once more improved by constructing the semantic representation in parallel with the syntactic analysis. Hereby, the number of possible interpretations will be reduced at an early point of analysis which increases the efficiency of the parser significantly. We believe that the proposed approach constitutes an important application to deductive databases as well as a promising direction towards sucessful natural language interface design.

6. Acknowledgement

Our work is supported by the Jubiläumsfonds of the Oesterreichische Nationalbank - research project no. 4147: *Natural Language Interface to Deductive Databases.*

References

[1] W. Winiwarter & A M. Tjoa. Natural Language Interfaces as Integrated Constituents of Deductive Databases. *NDA-93.*

[2] S. Naqvi & S. Tsur. *A Logical Language for Data and Knowledge Bases*, Rockville: Computer Science Press, 1989.

[3] D. Chimenti, R. Gamboa, R. Krishnamurthy, S. Naqvi, S. Tsur & C. Zaniolo. The LDL System Prototype. *IEEE Trans. on Knowledge and Data, Vol. 2, No. 1, 1990.*

[4] W. Winiwarter & A M. Tjoa. Morphological Analysis in Integrated Natural Language Interfaces to Deductive Databases. *NLULP4*, 1993.

[5] M. Bates. Natural-Language Interfaces. In *Encyclopaedia of Artificial Intelligence,* New York: Wiley, 1987.

[6] Y. Bar-Hillel. On Categorial and Phrase Structure Grammars. In *Language and Information,* Reading: Addison-Wesley, 1964.

[7] M. Steedman. Dependency and Coordination in the Grammar of Dutch and English, *Language,* Vol. 61, 1985.

[8] M. Steedman. Combinatory Grammars and Parasitic Gaps. *Natural Language and Linguistic Theory*, Vol. 5, 1987.

[9] D. Dowty. Type Raising, Functional Composition, and Non-Constituent Conjunction. In *Categorial Grammar and Natural Language Structures,* Dordrecht: Reidel, 1987.

[10] D. Weir & A. Joshi. Combinatory Categorial Grammars: Generative Power and Relationship to Linear Context-Free Rewriting System. *ACL-88.*

[11] H. Zeevat. Combining Categorial Grammar and Unification. In *Natural Language Processing and Linguistic Theories,* Dordrecht: Reidel, 1988.

[12] B. Hoffman. The Formal Consequences of Using Variables in CCG Categories. *ACL-93.*

[13] R. Seiffert. STUF: Ein flexibler Graphunifikationsformalismus und seine Anwendung in LILOG (in German). In *Wissensbasierte Systeme,* Berlin: Springer, 1988.

[14] B. Carpenter. The Generative Power of Categorial Grammars and Head-Driven Phrase Structure Grammars with Lexical Rules. *Comp. Linguistics,* Vol. 17, No. 3, 1991.

[15] K. Wittenburg. *Natural Language Parsing with Combinatory Categorial Grammar in a Graph-Unification Based Formalism*, Diss., Austin: Univ. Texas, 1986.

[16] K. Wittenburg. Predictive Combinators: A Method for Efficient Parsing of Combinatory Categorial Grammars. *ACL-87.*

[17] K. Wittenburg & R.E. Wall. Parsing with Categorial Grammar in Predictive Normal Form. In *Current Issues in Parsing Technology,* Boston: Kluwer, 1991.

[18] E. König. Parsing as Natural Deduction, *ACL-89.*

[19] S.M. Shieber, Y. Schabes & F.C.N. Pereira. *Principles and Implementations of Deductive Parsing,* Tech. Rep., TR-11-94, Harvard Univ., 1994.

[20] R. Pareschi & M. Steedman. A Lazy Way to Chart-Parse with Categorial Grammars. *ACL-87.*

[21] K. Vijay-Shanker & D.J. Weir. Polynomial Parsing of Combinatory Categorial Grammars. *ACL-90.*

[22] K. Vijay-Shanker & D.J. Weir. Parsing Some Constraint Grammar Formalisms. *Comp. Linguistics,* Vol. 19, No. 4, 1994.

[23] H. Uszkoreit. Categorial Unification Grammars. *COLING-86.*

[24] J. Hoeksema. *Categorial Morphology,* Diss., Univ. Groningen, 1984.

[25] P.J. Whitelock. A Feature-Based Categorial Morpho-Syntax for Japanese. In *Natural Language Parsing and Linguistic Theories,* Dordrecht: Reidel, 1988.

[26] H.-J. Novak & B. Wesche. Analyse und Synthese in einer kategorialen Unifikationsgrammatik: Möglichkeiten und Grenzen (in German), *KI-89.*

[27] M. Steedman. Parsing Spoken Language. In *Current Issues in Parsing Technology,* Boston: Kluwer, 1991.

[28] G. Bouma. Modifiers and Specifiers in Categorial Unification Grammar. *Comp. Linguistics,* Vol. 26, 1988.

[29] P. Maier, P. Steffens. Determinatoren und Quantoren in einer kategorialen Unifikationsgrammatik des Deutschen (in German). *ÖAIT-88.*

[30] B. Wesche. Non-Constituent Coordination ohne Funktionale Komposition und Typenanhebung (in German). *ÖAIT-88.*

[31] S. Millies. Kategoriales Parsing mit definiten Klauseln (in German). *KI-89.*

[32] C. Hauenschild. GPSG and German Word Order. In *Natural Language Parsing and Linguistic Theories,* Dordrecht: Reidel, 1988.

[33] S. Meknavin, T. Theeramunkong, H. Tanaka. Parsing Ill-Formed Input with ID/LP-Rules. *NLULP4*, 1993.